CPP05 – EX02
Soutenance Oral : Questions / Réponses pour Débutant

I. Introduction (présentation 2 minutes)

Cet exercice introduit les classes abstraites, l'héritage, le polymorphisme et l'exécution d'actions via un système de formulaires. La classe AForm définit les règles communes : nom, signature, grade requis pour signer et exécuter. Elle contient une méthode virtuelle pure executeAction(), ce qui la rend abstraite.

Trois classes concrètes héritent de AForm :
- ShrubberyCreationForm (création d'un fichier ASCII),
- RobotomyRequestForm (robotomisation 50 % réussite),
- PresidentialPardonForm (pardon présidentiel).

La méthode execute() d'AForm vérifie :
1. Le formulaire est signé,
2. Le bureaucrate a un grade suffisant.
Puis appelle executeAction().

II. Questions simples

Q1 : Pourquoi AForm est abstraite ?
R : Car elle contient une méthode virtuelle pure, donc non instanciable. Les formulaires concrets doivent implémenter executeAction().

Q2 : Différence entre execute() et executeAction() ?
R : execute() vérifie les règles communes (signature + grade). executeAction(), propre à chaque formulaire, réalise l'action concrète.

Q3 : Comment un formulaire devient signé ?
R : Via beSigned(), si le grade du bureaucrate est suffisant. Sinon, on lance une exception.

Q4 : Rôle des exceptions ?
R : Gérer les erreurs (grade trop bas, trop haut, non signé) proprement et éviter les codes
de retour complexes.

Q5 : Signer vs exécuter ?
R : Signer prépare, exécuter déclenche l'action. Les grades requis peuvent être différents.

Q6 : Rôle de executeForm() dans Bureaucrat ?
R : Envelopper un appel à execute() dans un try/catch et afficher un message clair.

Q7 : Pourquoi 1 à 150 ?
R : Règle du sujet. 1 est le meilleur grade, 150 le plus bas.

III. Questions sur les formulaires

Q8 : ShrubberyCreationForm ?

R : Crée un fichier <target>_shrubbery avec des arbres ASCII.

Q9 : RobotomyRequestForm ?
R : Robotomisation 1 fois sur 2. Semi■aléatoire.

Q10 : PresidentialPardonForm ?
R : Affiche que la cible a été pardonnée par Zaphod Beeblebrox.

IV. Questions plus avancées

Q11 : Pourquoi les attributs d'AForm sont privés ?
R : Pour garantir l'encapsulation et éviter que les classes dérivées modifient son état interne.

Q12 : Pourquoi séparer execute() et executeAction() ?
R : C'est un template method pattern. La logique commune est dans execute(), la logique spécifique dans executeAction().

Q13 : Pourquoi ne pas tout mettre dans les formulaires dérivés ?
R : Pour éviter la duplication et centraliser la logique des règles (signature + grade).

Q14 : Pourquoi pas "using namespace std" ?
R : Interdit par le sujet. Pour éviter les collisions de noms et les mauvaises pratiques.

V. Conclusion

Cet exercice montre comment structurer un système orienté objet avec une classe de base abstraite, des classes dérivées spécialisées, du polymorphisme et des exceptions. Il permet
de manipuler des objets via une interface commune (AForm) tout en personnalisant leur comportement dans executeAction().